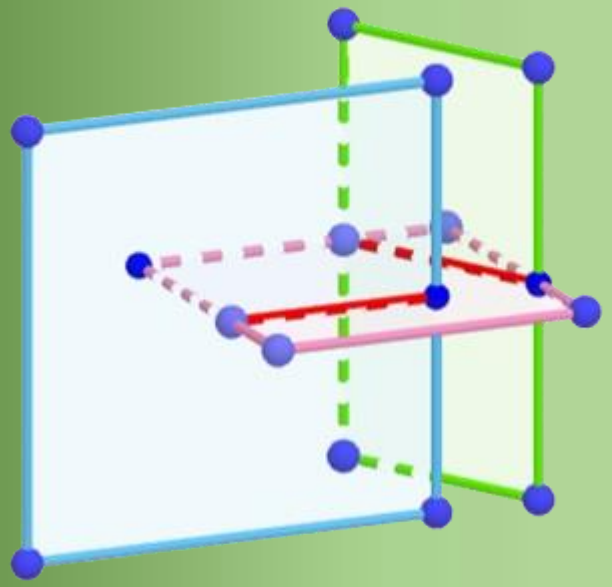
Politecnico di Torino



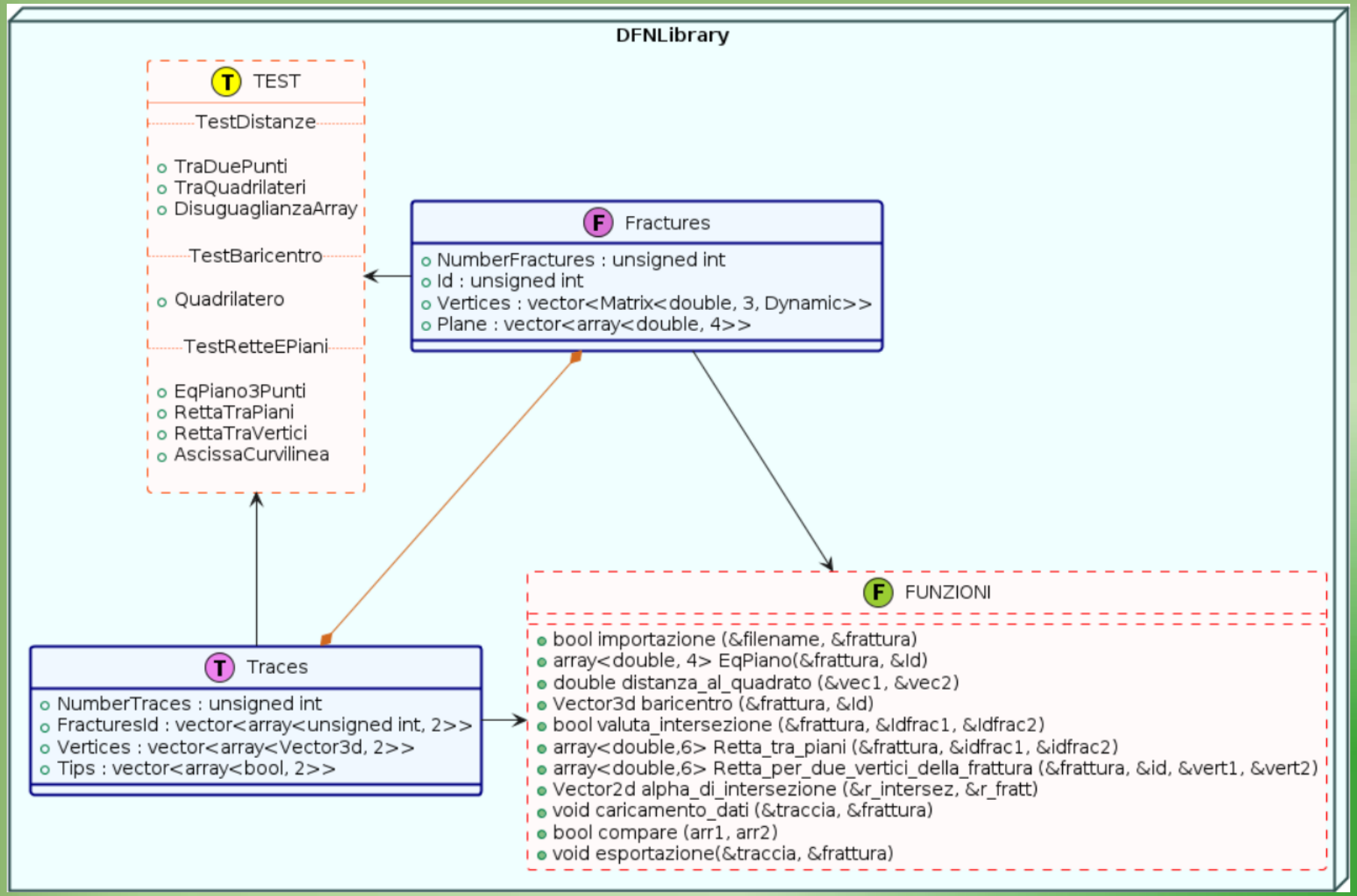
DISCRETE FRACTURES NETWORK

Marco Buffo Blin - 297583

Giulia Calabrese - 294808

Sofia Silvestro - 281693

STRUTTURE DATI



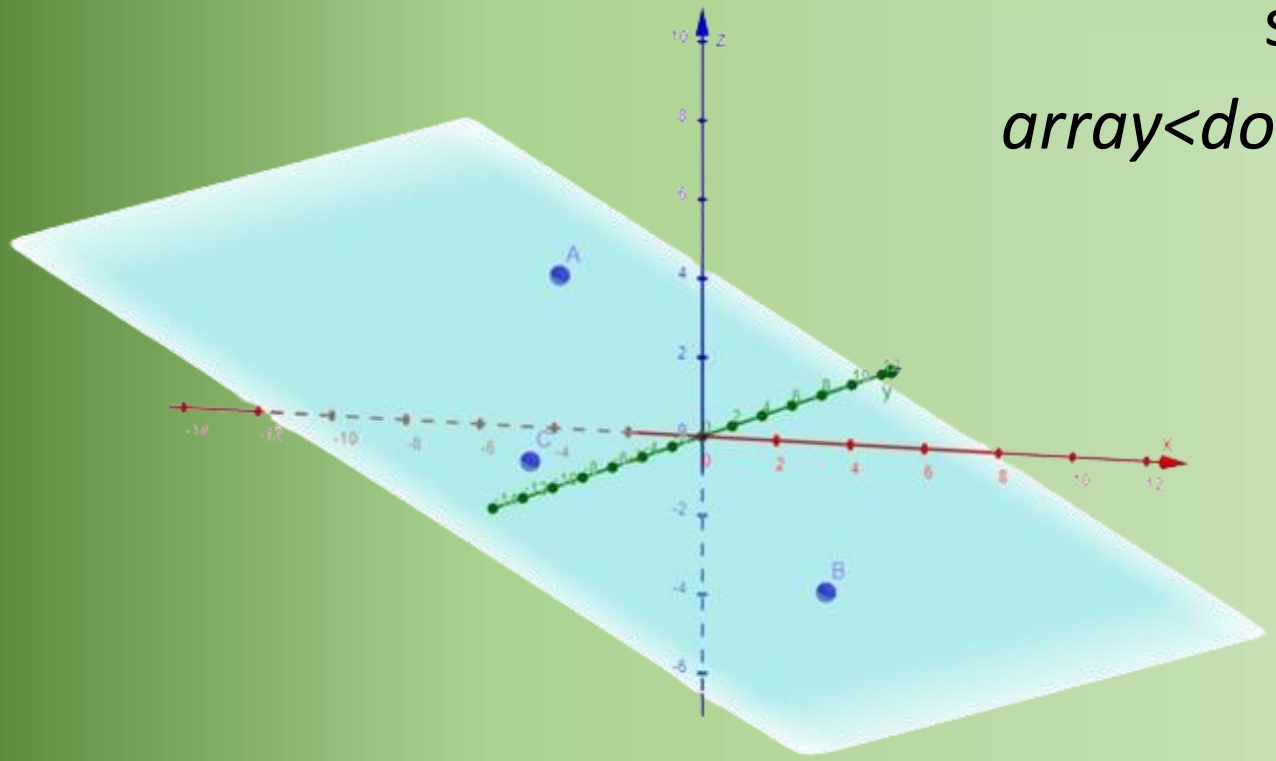
FUNZIONI IMPLEMENTATE

1. Import per lettura dati da file: *bool importazione(& filename, & frattura)*

2. Calcolo dell'equazione del piano su cui giace ogni frattura:

array<double, 4> EqPiano(& frattura, & Id)

$$ax + by + cz + d = 0$$

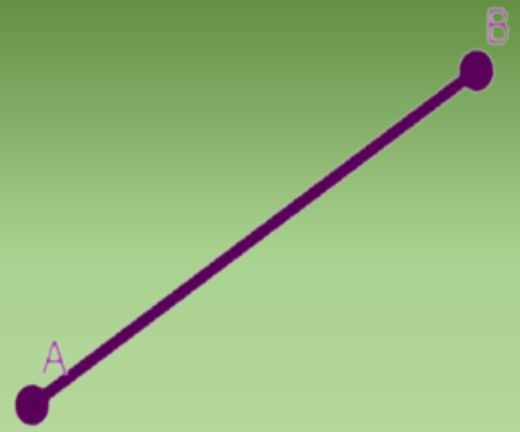


$$A = \begin{bmatrix} i & j & k \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{bmatrix}$$

$$a = (y_2 - y_1) \cdot (z_3 - z_1) - (y_3 - y_1) \cdot (z_2 - z_1)$$
$$b = -((x_2 - x_1) \cdot (z_3 - z_1) - (x_3 - x_1) \cdot (z_2 - z_1))$$
$$c = (x_2 - x_1) \cdot (y_3 - y_1) - (x_3 - x_1) \cdot (y_2 - y_1)$$
$$d = -a \cdot x_1 - b \cdot y_1 - c \cdot z_1$$

3. Distanza euclidea al quadrato tra due vettori di dimensione 3:

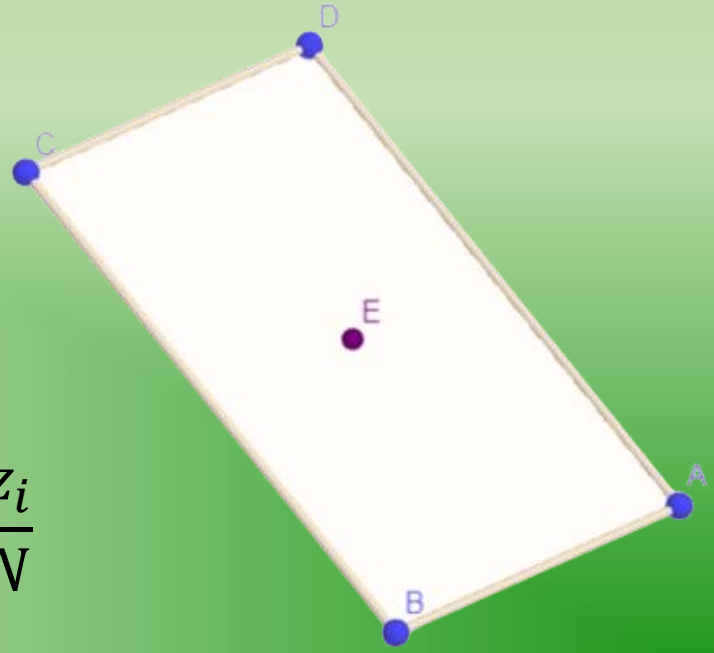
double distanza_al_quadrato(& vec1, & vec2)



$$d^2 = (x_1 - x_2) \cdot (x_1 - x_2) + (y_1 - y_2) \cdot (y_1 - y_2) + (z_1 - z_2) \cdot (z_1 - z_2)$$

4. Calcolo del baricentro di un poligono:

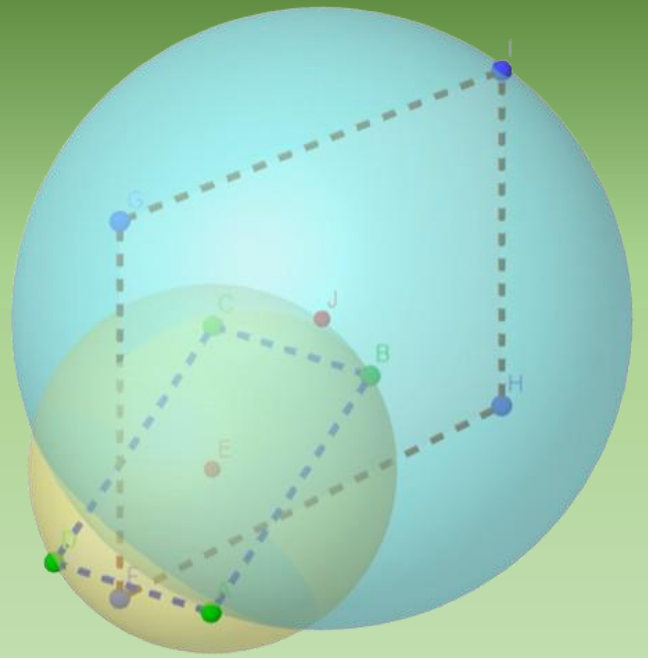
Vector3d baricentro (& frattura, & Id1)



$$x_G = \sum_{i=0}^{N} \frac{x_i}{N} \qquad y_G = \sum_{i=0}^{N} \frac{y_i}{N} \qquad z_G = \sum_{i=0}^{N} \frac{z_i}{N}$$

5. Determina se due fratture potrebbero intersecarsi con calcolo distanza tra baricentri e sfere circoscritte:

bool valuta_intersezione (& frattura, & Idfrac1, & Idfrac2)



6. Retta di intersezione tra due fratture:

array<double,6> (& frattura, & idfrac1, & idfrac2)

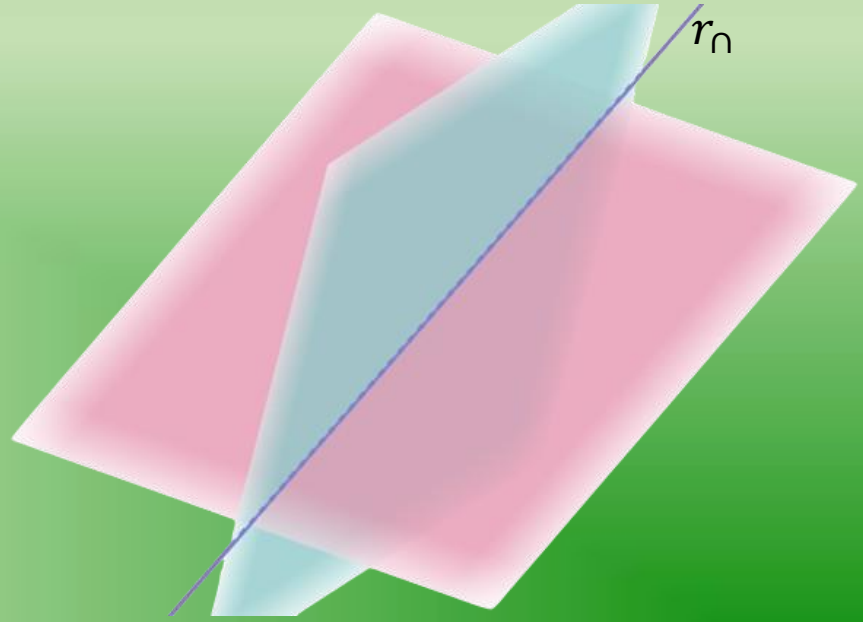
$$r_{\cap}: \begin{cases} x = a \cdot t + d \\ y = b \cdot t + e \\ z = c \cdot t + f \end{cases}$$

$$P_1: a_1x + b_1y + c_1z + d_1 = 0$$

$$P_2: a_2x + b_2y + c_2z + d_2 = 0$$

$$M = \begin{bmatrix} x & y & z \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{bmatrix}$$

$$a = (b_1 \cdot c_2 - b_2 \cdot c_1)$$
$$b = -(a_1 \cdot c_2 - a_2 \cdot c_1)$$
$$c = (a_1 \cdot b_2 - a_2 \cdot b_1)$$



$$\mathrm{A} \cdot \mathrm{x} = \mathrm{b} \quad \rightarrow \quad A = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a & b & c \end{bmatrix} \qquad x = \begin{bmatrix} d \\ e \\ f \end{bmatrix} \qquad b = \begin{bmatrix} -d_1 \\ -d_2 \\ 0 \end{bmatrix}$$

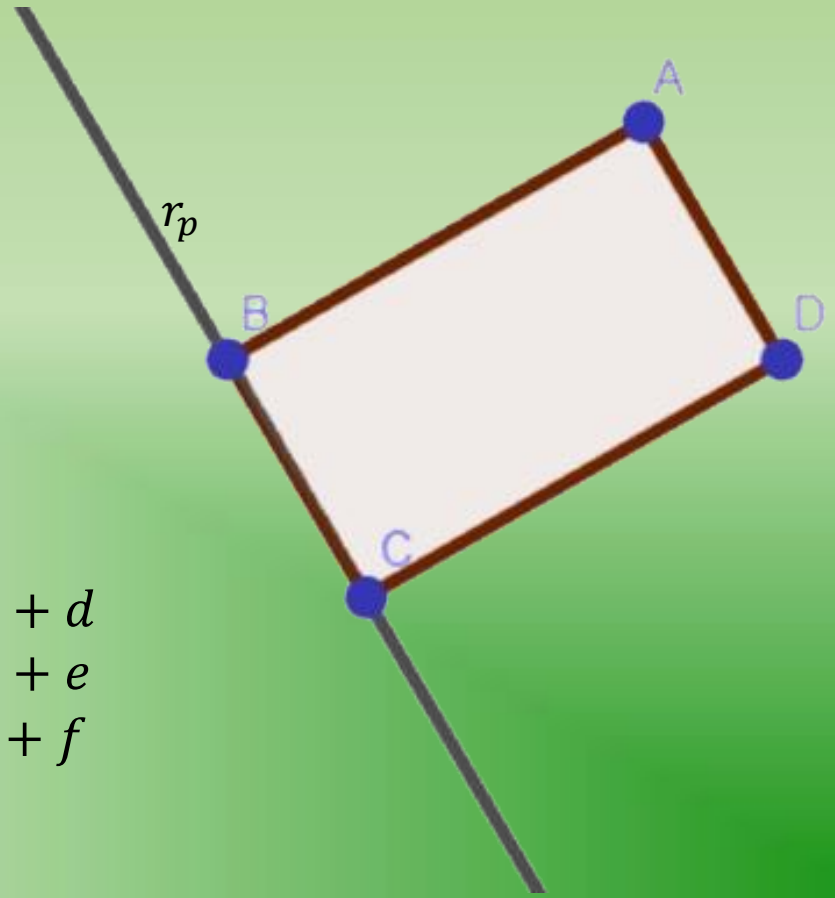
7. Retta passante per due vertici di una frattura:

array<double,6> Retta_per_due_vertici_della_frattura(& frattura, & id, & vert1, & vert2)

$$a = (x_2 - x_1)$$
$$b = (y_2 - y_1)$$
$$c = (z_2 - z_1)$$

$$V_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} d \\ e \\ f \end{bmatrix} \qquad V_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

$$r_p: \begin{cases} x = a \cdot t + d \\ y = b \cdot t + e \\ z = c \cdot t + f \end{cases}$$



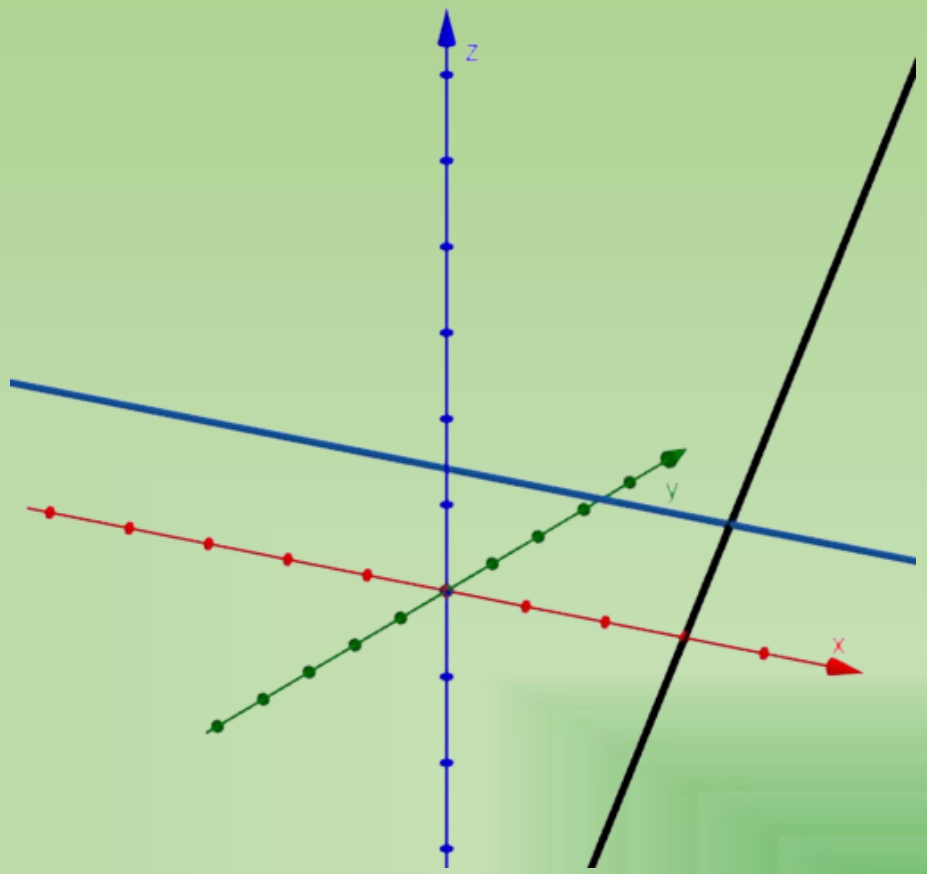
8. Calcolo delle ascisse curvilinee delle rette che si intersecano per trovare un punto di intersezione:

Vector2d alpha_di_intersezione(& r_intersez, & r_fratt)

$$r_{\cap}: \begin{cases} x = a_{\cap} \cdot t + d_{\cap} \\ y = b_{\cap} \cdot t + e_{\cap} \\ z = c_{\cap} \cdot t + f_{\cap} \end{cases} \qquad r_p: \begin{cases} x = a \cdot t + d \\ y = b \cdot t + e \\ z = c \cdot t + f \end{cases}$$

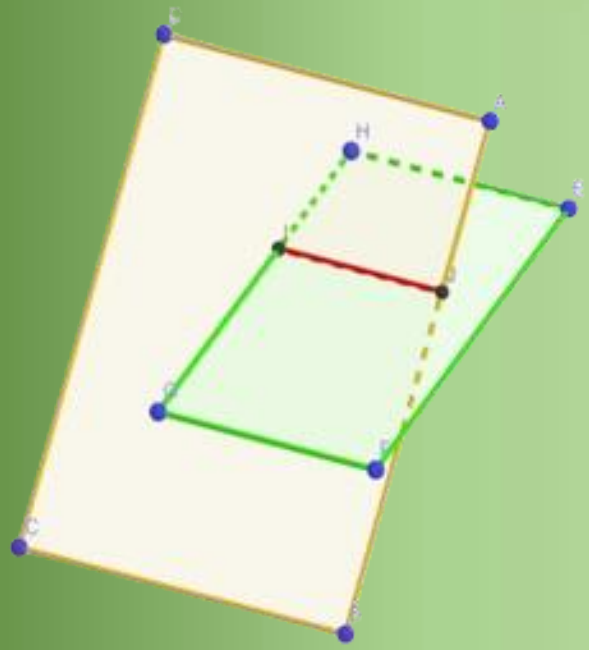
$$A \cdot x = b$$

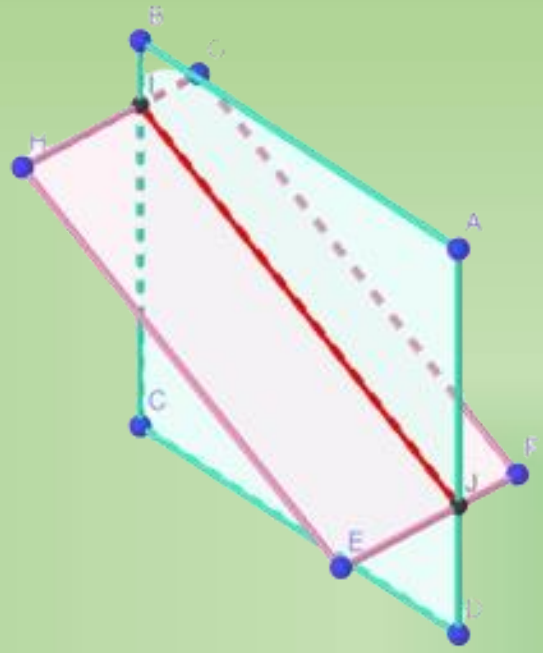
$$A = \begin{bmatrix} a_p & -a_{\cap} \\ b_p & -b_{\cap} \\ c_p & -c_{\cap} \end{bmatrix} \qquad x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \qquad b = \begin{bmatrix} d_{\cap} - d_p \\ e_{\cap} - e_p \\ f_{\cap} - f_p \end{bmatrix}$$

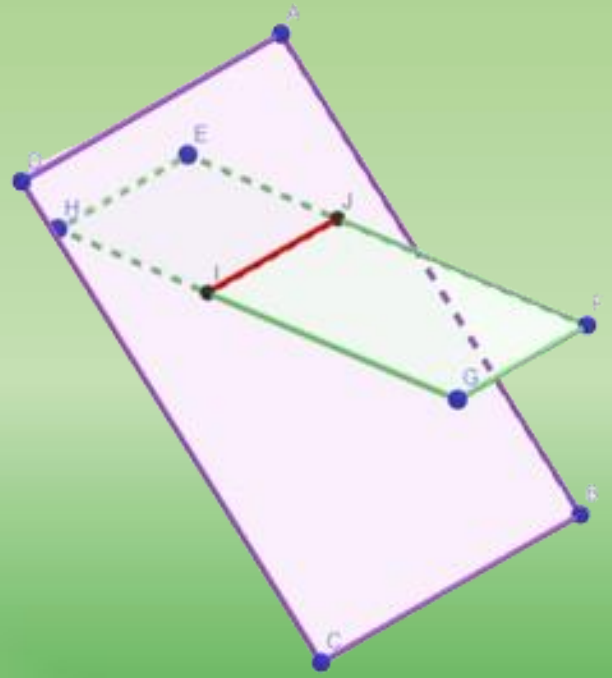


9. Caricamento dei dati nella struttura:

void caricamento_dati(& traccia, & frattura)







10. Ordinamento delle tracce:

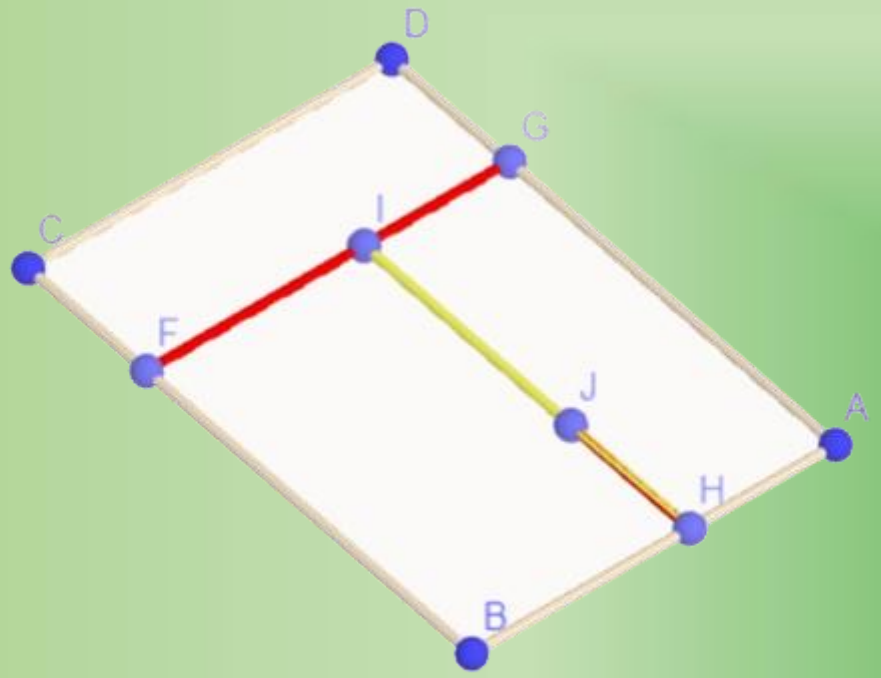
bool compare(arr1, arr2)

11. Esportazione dei dati in un file di testo:

void esportazione(& traccia, & frattura)

12. Seconda parte:

void caricamento_dati_2(& traccia, & frattura, & mesh)



TEST DI UNITÁ

Implementazione dei GoogleTest con la funzione ASSERT e alcune sue derivate:

- Applicazione delle formule matematiche
- Esecuzione prove pratiche su carta
- Confronto tra il risultato dato dalla funzione e il risultato ricavato a mano.

CONCLUSIONI

- Divisione del progetto in tre parti
- Aspetti computazionali di accesso ai dati nelle strutture dati utilizzate
- Il valore utilizzato per la tolleranza è 10^{-10}
- I diversi metodi utilizzati per la risoluzione dei sistemi lineari hanno costi computazionali diversi e sono stati scelti in base alle caratteristiche delle matrici su cui si stava lavorando